Assessing the Ability of Self-Attention Networks to Learn Word Order

Baosong Yang[†] **Longyue Wang**[‡] **Derek F. Wong**[†] **Lidia S. Chao**[†] **Zhaopeng Tu**[‡*]
[†]NLP^{2}CT Lab, Department of Computer and Information Science, University of Macau
nlp2ct.baosong@gmail.com, {derekfw,lidiasc}@umac.mo
[‡]Tencent AI Lab
{vinnylywang,zptu}@tencent.com

Abstract

Self-attention networks (SAN) have attracted a lot of interests due to their high parallelization and strong performance on a variety of NLP tasks, e.g. machine translation. Due to the lack of recurrence structure such as recurrent neural networks (RNN), SAN is ascribed to be weak at learning positional information of words for sequence modeling. However, neither this speculation has been empirically confirmed, nor explanations for their strong performances on machine translation tasks when "lacking positional information" have been explored. To this end, we propose a novel *word reordering detection* task to quantify how well the word order information learned by SAN and RNN. Specifically, we randomly move one word to another position, and examine whether a trained model can detect both the original and inserted positions. Experimental results reveal that: 1) SAN trained on *word reordering detection* indeed has difficulty learning the positional information even with the position embedding; and 2) SAN trained on *machine translation* learns better positional information than its RNN counterpart, in which position embedding plays a critical role. Although recurrence structure make the model more universally-effective on learning word order, learning objectives matter more in the downstream tasks such as machine translation.

1 Introduction

Self-attention networks (SAN, Parikh et al., 2016; Lin et al., 2017) have shown promising empirical results in a variety of natural language processing (NLP) tasks, such as machine translation (Vaswani et al., 2017), semantic role labelling (Strubell et al., 2018), and language representations (Devlin et al., 2019). The popularity of SAN lies in its high parallelization in computation, and flexibility in modeling dependencies regardless of distance by explicitly attending to all the signals. Position embedding (Gehring et al., 2017) is generally deployed to capture sequential information for SAN (Vaswani et al., 2017; Shaw et al., 2018).

Recent studies claimed that SAN with position embedding is still weak at learning word order information, due to the lack of recurrence structure that is essential for sequence modeling (Shen et al., 2018a; Chen et al., 2018; Hao et al., 2019). However, such claims are mainly based on a theoretical argument, which have not been empirically validated. In addition, this can not explain well why SAN-based models outperform their RNN counterpart in machine translation – a benchmark sequence modeling task (Vaswani et al., 2017).

Our goal in this work is to empirically assess the ability of SAN to learn word order. We focus on asking the following research questions:

Q1: Is recurrence structure obligate for learning word order, and does the conclusion hold in different scenarios (e.g., translation)?

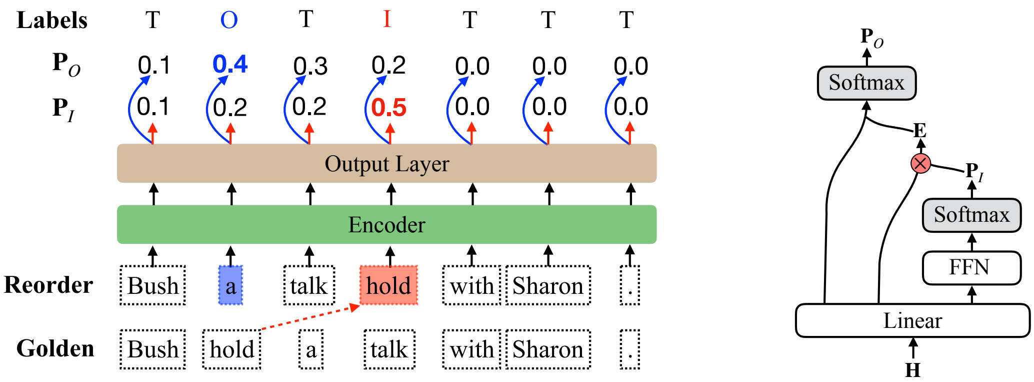
Q2: Is the model architecture the critical factor for learning word order in the downstream tasks such as machine translation?

Q3: Is position embedding powerful enough to capture word order information for SAN?

We approach these questions with a novel probing task – *word reordering detection* (WRD), which aims to detect the positions of randomly reordered words in the input sentence. We compare SAN with RNN, as well as directional SAN (DiSAN, Shen et al., 2018a) that augments SAN with recurrence modeling. In this study, we focus on the encoders implemented with different architectures, so as to investigate their abilities to learn

[*] Zhaopeng Tu is the corresponding author of the paper. This work was conducted when Baosong Yang was interning at Tencent AI Lab.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3635–3644
Florence, Italy, July 28 - August 2, 2019. ©2019 Association for Computational Linguistics



(a) Position Detector

(b) Output Layer

Figure 1: Illustration of (a) the position detector, where (b) the output layer is build upon a randomly initialized or pre-trained encoder. In this example, the word "hold" is moved to another place. The goal of this task is to predict the inserted position "I" and the original position "O" of "hold".

word order information of the input sequence. The encoders are trained on objectives like detection accuracy and machine translation, to study the influences of learning objectives.

Our experimental results reveal that: (Q1) SAN indeed underperforms the architectures with recurrence modeling (i.e. RNN and DiSAN) on the WRD task, while this conclusion does not hold in machine translation: SAN trained with the translation objective outperforms both RNN and DiSAN on detection accuracy; (Q2) Learning objectives matter more than model architectures in downstream tasks such as machine translation; and (Q3) Position encoding is good enough for SAN in machine translation, while DiSAN is a more universally-effective mechanism to learn word order information for SAN.

Contributions The key contributions are:

- We design a novel probing task along with the corresponding benchmark model, which can assess the abilities of different architectures to learn word order information.[1]

- Our study dispels the doubt on the inability of SAN to learn word order information in machine translation, indicating that the learning objective can greatly influence the suitability of an architecture for downstream tasks.

2 Word Reordering Detection Task

In order to investigate the ability of self-attention networks to extract word order information, in this section, we design an artificial task to evaluate the abilities of the examined models to detect the erroneous word orders in a given sequence.

Task Description Given a sentence $X = \{x_1, ..., x_i, ..., x_N\}$, we randomly pop a word x_i and insert it into another position j ($1 \leq i, j \leq N$ and $i \neq j$). The objective of this task is to detect both the position the word is popped out (labeled as "O"), as well as the position the word is inserted (labeled as "I"). As seen the example in Figure 1 (a), the word "hold" is moved from the 2^{nd} slot to the 4^{th} slot. Accordingly, the 2^{nd} and 4^{th} slots are labelled as "O" and "I", respectively. To exactly detect word reordering, the examined models have to learn to recognize both the normal and abnormal word order in a sentence.

Position Detector Figure 1 (a) depicts the architecture of the position detector. Let the sequential representations $\mathbf{H} = \{\mathbf{h}_1, ..., \mathbf{h}_N\}$ be the output of each encoder noted in Section 3, which are fed to the output layer (Figure 1 (b)). Since only one pair of "I" and "O" labels should be generated in the output sequence, we cast the task as a pointer detection problem (Vinyals et al., 2015). To this end, we turn to an output layer that commonly used in the reading comprehension task (Wang and Jiang, 2017; Du and Cardie, 2017), which aims to identify the start and end positions of the answer in the given text.[2] The output layer consists of two sub-layers, which progressively predicts the prob-

[1] The data and codes are released at: https://github.com/baosongyang/WRD.

[2] Contrary to reading comprehension in which the start and end positions are ordered, "I" and "O" do not have to be ordered in our tasks, that is, the popped word can be inserted to either left or right position.

abilities of each position being labelled as "I" and "O". The probability distribution of the sequence being labelled as "I" is calculated as:

$$\mathbf{P}_I = \texttt{SoftMax}(\mathbf{U}_I^\top \texttt{tanh}(\mathbf{W}_I \mathbf{H})) \quad \in \mathbb{R}^N \quad (1)$$

where $\mathbf{W}_I \in \mathbb{R}^{d \times d}$ and $\mathbf{U}_I \in \mathbb{R}^d$ are trainable parameters, and d is the dimensionality of $\mathbf{H}$.

The second layer aims to locate the original position "O", which conditions on the predicted popped word at the position "I".[3] To make the learning process differentiable, we follow Xu et al. (2017) to use the weighted sum of hidden states as the approximate embedding $\mathbf{E}$ of the popped word. The embedding subsequently serves as a query to attend to the sequence $\mathbf{H}$ to find which position is most similar to the original position of popped word. The probability distribution of the sequence being labelled as "O" is calculated as:

$$\mathbf{E} = \mathbf{P}_I(\mathbf{W}_Q \mathbf{H}) \qquad \in \mathbb{R}^d \qquad (2)$$

$$\mathbf{P}_O = \textsc{Att}(\mathbf{E}, \mathbf{W}_K \mathbf{H}) \qquad \in \mathbb{R}^N \qquad (3)$$

where $\{\mathbf{W}_Q, \mathbf{W}_K\} \in \mathbb{R}^{d \times d}$ are trainable parameters that transform $\mathbf{H}$ to query and key spaces respectively. $\textsc{Att}(\cdot)$ denotes the dot-product attention (Luong et al., 2015; Vaswani et al., 2017).

Training and Predicting In training process, the objective is to minimize the cross entropy of the true inserted and original positions, which is the sum of the negative log probabilities of the groundtruth indices by the predicted distributions:

$$L = \mathbf{Q}_I^\top \log \mathbf{P}_I + \mathbf{Q}_O^\top \log \mathbf{P}_O \qquad (4)$$

where $\{\mathbf{Q}_I, \mathbf{Q}_O\} \in \mathbb{R}^N$ is an one-hot vector to indicate the groundtruth indices for the inserted and original positions. During prediction, we choose the positions with highest probabilities from the distributions $\mathbf{P}_I$ and $\mathbf{P}_O$ as "I" and "O", respectively. Considering the instance in Figure 1 (a), the 4^{th} position is labelled as inserted position "I", and the 2^{nd} position as the original position "O".

3 Experimental Setup

In this study, we strove to empirically test whether SAN indeed weak at learning positional information and come up with the reason about the strong performance of SAN on machine translation. In response to the three research questions in Section 1, we give following experimental settings:

- Q1: We compare SAN with two recurrence architectures – RNN and DiSAN on the WRD task, thus to quantify their abilities on learning word order (Section 3.1).
- Q2: To compare the effects of learning objectives and model architectures, we train each encoder under two scenarios, i.e. trained on objectives like WRD accuracy and on machine translation (Section 3.2).
- Q3: The strength of position encoding is appraised by ablating position encoding and recurrence modeling for SAN.

3.1 Encoder Setting

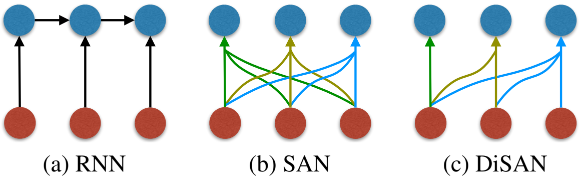


Figure 2: Illustration of (a) RNN; (b) SAN; and (c) DiSAN. Colored arrows denote parallel operations.

RNN and SAN are commonly used to produce sentence representations on NLP tasks (Cho et al., 2014; Lin et al., 2017; Chen et al., 2018). As shown in Figure 2, we investigate three architectures in this study. Mathematically, let $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\} \in \mathbb{R}^{d \times N}$ be the embedding matrix of the input sentence, and $\mathbf{H} = \{\mathbf{h}_1, \ldots, \mathbf{h}_N\} \in \mathbb{R}^{d \times N}$ be the output sequence of representations.

- **RNN** sequentially produces each state:

  $$\mathbf{h}_n = f(\mathbf{h}_{n-1}, \mathbf{x}_n), \qquad (5)$$

  where $f(\cdot)$ is GRU (Cho et al., 2014) in this study. RNN is particularly hard to parallelize due to their inherent dependence on the previous state $\mathbf{h}_{n-1}$.

- **SAN** (Lin et al., 2017) produces each hidden state in a parallel fashion:

  $$\mathbf{h}_n = \textsc{Att}(\mathbf{q}_n, \mathbf{K})\mathbf{V}, \qquad (6)$$

  where the query $\mathbf{q}_n \in \mathbb{R}^d$ and the keys and values $(\mathbf{K}, \mathbf{V}) \in \mathbb{R}^{d \times N}$ are transformed from $\mathbf{X}$. To imitate the order of the sequence, Vaswani et al. (2017) deployed position encodings (Gehring et al., 2017) into SAN.

[3] We tried to predict the position of "O" without feeding the approximate embedding, i.e. predicting "I" and "O" individually. It slightly underperforms the current model.

- **DiSAN** (Shen et al., 2018a) augments SAN with the ability to encode word order:

$$\mathbf{h}_n = \textsc{Att}(\mathbf{q}_n, \mathbf{K}_{\leq n})\mathbf{V}_{\leq n}, \tag{7}$$

where $(\mathbf{K}_{\leq n}, \mathbf{V}_{\leq n})$ indicate leftward elements, e.g., $\mathbf{K}_{\leq n} = \{\mathbf{k}_1, \dots, \mathbf{k}_n\}$.

To enable a fair comparison of architectures, we only vary the sub-layer of sequence modeling (e.g. the SAN sub-layer) in the Transformer encoder (Vaswani et al., 2017), and keep the other components the same for all architectures. We use bi-directional setting for RNN and DiSAN, and apply position embedding for SAN and DiSAN. We follow Vaswani et al. (2017) to set the configurations of the encoders, which consists of 6 stacked layers with the layer size being 512.

3.2 Learning Objectives

In this study, we employ two strategies to train the encoders, which differ at the learning objectives and data used to train the associated parameters. Note that in both strategies, the output layer in Figure 2 is fine-trained on the WRD data with the word reordering detection objective.

WRD Encoders We first directly train the encoders on the WRD data, to evaluate the abilities of model architectures. The WRD encoders are randomly initialized and co-trained with the output layer. Accordingly, the detection accuracy can be treated as the learning objective of this group of encoders. Meanwhile, we can investigate the reliability of the proposed WRD task by checking whether the performances of different architectures (i.e. RNN, SAN, and DiSAN) are consistent with previous findings on other benchmark NLP tasks (Shen et al., 2018a; Tang et al., 2018; Tran et al., 2018; Devlin et al., 2019).

NMT Encoders To quantify how well different architectures learn word order information with the learning objective of machine translation, we first train the NMT models (both encoder and decoder) on bilingual corpus using the same configuration reported by Vaswani et al. (2017). Then, we *fix the parameters of the encoder*, and only train the parameter associated with the output layer on the WRD data. In this way, we can probe the representations learned by NMT models, on their abilities to learn word order of input sentences.

To cope with WRD task, all the models were trained for 600K steps, each of which is allocated a batch of 500 sentences. The training set is shuffled after each epoch. We use Adam (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 10^{-9}$. The learning rate linearly warms up over the first 4,000 steps, and decreases thereafter proportionally to the inverse square root of the step number. We use a dropout rate of 0.1 on all layers.

3.3 Data

Machine Translation We pre-train NMT models on the benchmark WMT14 English⇒German (En⇒De) data, which consists of 4.5M sentence pairs. The validation and test sets are newstest2013 and newstest2014, respectively. To demonstrate the universality of the findings in this study, we also conduct experiments on WAT17 English⇒Japanese (En⇒Ja) data. Specifically, we follow Morishita et al. (2017) to use the first two sections of WAT17 dataset as the training data, which approximately consists of 2.0M sentence pairs. We use newsdev2017 as the validation set and newstest2017 as the test set.

Word Reordering Detection We conduct this task on the English sentences, which are extracted from the source side of WMT14 En⇒De data with maximum length to 80. For each sentence in different sets (i.e. training, validation, and test sets), we construct an instance by randomly moving a word to another position. Finally we construct 7M, 10K and 10K samples for training, validating and testing, respectively. Note that a sentence can be sampled multiple times, thus each dataset in the WRD data contains more instances than that in the machine translation data.

All the English and German data are tokenized using the scripts in Moses. The Japanese sentences are segmented by the word segmentation toolkit KeTea (Neubig et al., 2011). To reduce the vocabulary size, all the sentences are processed by byte-pair encoding (BPE) (Sennrich et al., 2016) with 32K merge operations for all the data.

4 Experimental Results

We return to the central questions originally posed, that is, whether SAN is indeed weak at learning positional information. Using the above experimental design, we give the following answers:

A1: SAN-based encoder trained on the WRD data is indeed harder to learn positional information than the recurrence architectures (Section 4.1), while there is no evidence that

Models	Insert	Original	Both
RNN	78.4	**73.4**	**68.2**
SAN	73.2	66.0	60.1
DiSAN	**79.6**	70.1	68.0

Table 1: Accuracy on the WRD task. "Insert" and "Original" denotes the accuracies of detecting the inserted and original positions respectively, and "Both" denotes detecting both positions.

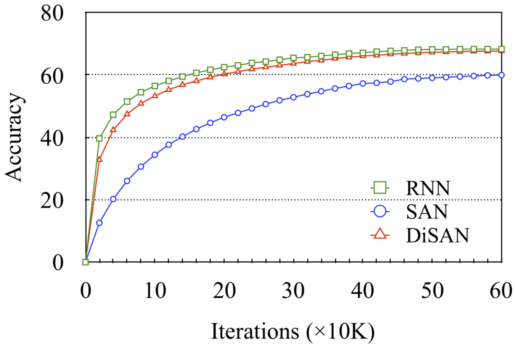


Figure 3: Learning curve of WRD encoders on WRD task. Y-axis denotes the accuracy on the validation set. Obviously, SAN has slower convergence.

SAN-based NMT encoders learns less word order information (Section 4.2);

A2: The learning objective plays a more crucial role on learning word order than the architecture in downstream tasks (Section 4.3);

A3: While the position encoding is powerful enough to capture word order information in machine translation, DiSAN is a more universally-effective mechanism (Table 2).

4.1 Results on WRD Encoders

We first check the performance of each WRD encoder on the proposed WRD task from two aspects: 1) WRD accuracy; and 2) learning ability.

WRD Accuracy The detection results are concluded in Table 1. As seen, both RNN and DiSAN significantly outperform SAN on our task, indicating that the recurrence structure (RNN) exactly performs better than parallelization (SAN) on capturing word order information in a sentence. Nevertheless, the drawback can be alleviated by applying directional attention functions. The comparable result between DiSAN and RNN confirms the hypothesis by Shen et al. (2018a) and Devlin et al. (2019) that directional SAN exactly improves the ability of SAN to learn word order. The consistency between prior studies and our results verified the reliability of the proposed WRD task.

Learning Curve We visualize the learning curve of the training. As shown in Figure 3, SAN has much slower convergence than others, showing that SAN has a harder time learning word order information than RNN and DiSAN. This is consistent with our intuition that the parallel structure is more difficult to learn word order information than those models with a sequential process. Considering DiSAN, although it has slightly slower learning speed at the early stage of the training, it is able to achieve comparable accuracy to RNN at the mid and late phases of the training.

4.2 Results on Pre-Trained NMT Encoders

We investigate whether the SAN indeed lacks the ability to learn word order information under machine translation context. The results are concluded in Table 2. We first report the effectiveness of the compared models on translation tasks. For En-De translation, SAN outperforms RNN, which is consistent with the results reported in (Chen et al., 2018). The tendency is universal on En-Ja which is a distant language pair (Bosch and Sebastián-Gallés, 2001; Isozaki et al., 2010). Moreover, DiSAN incrementally improves the translation quality, demonstrating that model directional information benefits to the translation quality. The consistent translation performances make the following evaluation on WRD accuracy convincing.

Concerning the performances of NMT encoders on the WRD task:

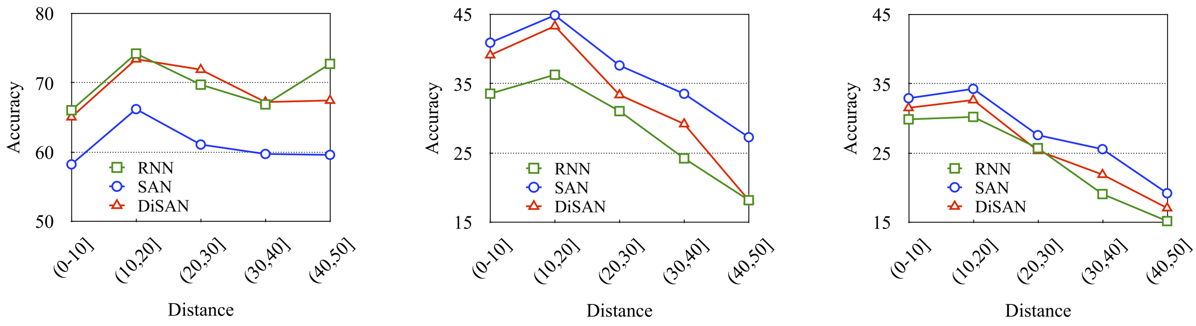
SAN-based NMT Encoder Performs Better It is surprising to see that SAN yields even higher accuracy on WRD task than other pre-trained NMT encoders, despite its lower translation qualities comparing with DiSAN. The results not only dispel the doubt on the inablity of SAN-based encoder to learn word order in machine translation, but also demonstrate that SAN learns to retain more features with respect to word order during the training of machine translation.

Learning Objectives Matter More In addition, both the NMT encoders underperform the WRD encoders on detection task across models and language pairs.[4] The only difference between the

[4] The En⇒Ja pre-trained encoders yield lower accuracy on WRD task than that of En⇒De pre-trained encoders. We

Model	Translation		Detection		
	En⇒De	En⇒Ja	En⇒De Enc.	En⇒Ja Enc.	WRD Enc.
RNN	26.8	42.9	33.9	29.0	**68.2**
SAN	27.3	43.6	**41.6**	**32.8**	60.1
- Pos_Emb	11.5	–	0.3	–	0.3
DiSAN	**27.6**	**43.7**	39.7	31.2	68.0
- Pos_Emb	27.0	43.1	40.1	31.0	62.8

Table 2: Performances of NMT encoders pre-trained on WMT14 En⇒De and WAT17 En⇒Ja data. “Translation” denotes translation quality measured in BLEU scores, while “Detection” denotes the accuracies on WRD task. “En⇒De Enc.” denotes NMT encoder trained with translation objective on the En⇒De data. We also list the detection accuracies of WRD encoders (“WRD Enc.”) for comparison. “- Pos_Emb” indicates removing positional embeddings from SAN- or DiSAN-based encoder. Surprisingly, *SAN-based NMT encoder achieves the best accuracy on the WRD task*, which contrasts with the performances of WRD encoders (the last column).



(a) WRD Encoder

(b) En⇒De NMT Encoder

(c) En⇒Ja NMT Encoder

Figure 4: Accuracy of pre-trained NMT encoders according to various distances between the positions of “O” and “I” (X-axis). As seen, the performance of each WRD encoder (a) is stable across various distances, while the pre-trained (b) En⇒De and (c) En⇒Ja encoders consistently get lower accuracy with the increasing of distance.

two kinds of encoders is the learning objective. This raises a hypothesis that the learning objective sometimes severs as a more critical factor than the model architecture on modeling word order.

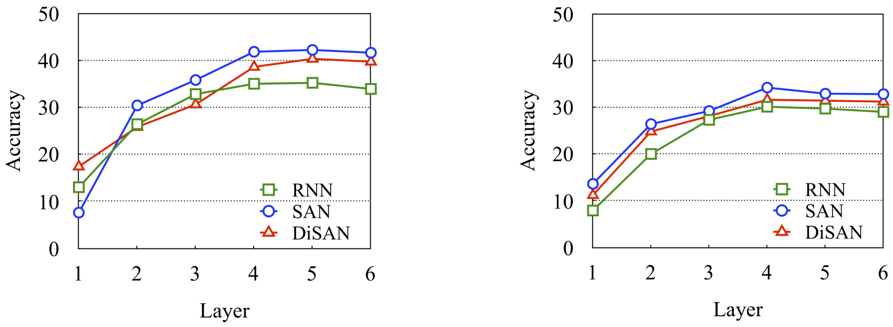
Position Encoding VS. Recurrence Modeling In order to assess the importance of position encoding, we redo the experiments by removing the position encoding from SAN and DiSAN (“-Pos_Emb”). Clearly, SAN-based encoder without position embedding fails on both machine translation and our WRD task, indicating the necessity of position encoding on learning word order. It is encourage to see that SAN yields higher BLEU score and detection accuracy than “DiSAN-Pos_Emb” in machine translation scenario. It means that position embedding is more suitable on capture word order information for machine translation than modeling recurrence for SAN. Considering both two scenarios, DiSAN-based encoders achieve comparable detection accuracies to the best models, revealing its effectiveness and universality on learning word order.

4.3 Analysis

In response to above results, we provide further analyses to verify our hypothesis on NMT encoders. We discuss three questions in this section: 1) Does learning objective indeed affect the extracting of word order information; 2) How SAN derives word order information from position encoding; and 3) Whether more word order information retained is useful for machine translation.

Accuracy According to Distance We further investigate the accuracy of WRD task according to various distance between the positions of word is popped out and inserted. As shown in Figure 4 (a), WRD encoders marginally reduce the performance with the increasing of distances. How-

attribute this to the difference between the source sentences in pre-training corpus (En-Ja) and that of WRD data (from En-De dataset). Despite of this, the tendency of results are consistent across language pairs.



(a) En⇒De NMT encoder

(b) En⇒Ja NMT encoder

Figure 5: Performance of each layer from (a) pre-trained En⇒De encoder and (b) pre-trained En⇒Ja encoder on WRD task. The evaluation are conducted on the test set. Clearly, the accuracy of SAN gradually increased with the stacking of layers and consistently outperform that of other models across layers.

ever, this kind of stability is destroyed when we pre-train each encoder with a learning objective of machine translation. As seen in Figure 4 (b) and (c), the performance of pre-trained NMT encoders obviously became worse on long-distance cases across language pairs and model variants. This is consistent with prior observation on NMT systems that both RNN and SAN fail to fully capture long-distance dependencies (Tai et al., 2015; Yang et al., 2017; Tang et al., 2018).

Regarding to information bottleneck principle (Tishby and Zaslavsky, 2015; Alemi et al., 2016), our NMT models are trained to maximally maintain the relevant information between source and target, while abandon irrelevant features in the source sentence, e.g. portion of word order information. Different NLP tasks have distinct requirements on linguistic information (Conneau et al., 2018). For machine translation, the local patterns (e.g. phrases) matter more (Luong et al., 2015; Yang et al., 2018, 2019), while long-distance word order information plays a relatively trivial role in understanding the meaning of a source sentence. Recent studies also pointed out that abandoning irrelevant features in source sentence benefits to some downstream NLP tasks (Lei et al., 2016; Yu et al., 2017; Shen et al., 2018b). An immediate consequence of such kind of data process inequality (Schumacher and Nielsen, 1996) is that information about word order that is lost in encoder cannot be recovered in the detector, and consequently drops the performance on our WRD task. The results verified that the learning objective indeed affects more on learning word order information than model architecture in our case.

Accuracy According to Layer Several researchers may doubt that the parallel structure of SAN may lead to failure on capturing word order information at higher layers, since the position embeddings are merely injected at the input layer. Accordingly, we further probe the representations at each layer on our WRD task to explore how does SAN learn word order information. As seen in Figure 5, SAN achieves better performance than other NMT encoders on the proposed WRD tasks across almost all the layers. The result dispels the doubt on the inability of position encoding and confirms the speculation by Vaswani et al. (2017) and Shaw et al. (2018) who suggested that SAN can profit from the use of residual network which propagates the positional information to higher layers. Moreover, both SAN and RNN gradually increase their performance on our task with the stacking of layers. The same tendency demonstrates that position encoding is able to provide same learning manner to that of recurrent structure with respect to word order for SAN. Both the results confirm the strength of position encoding to bring word order properties into SAN.

We strove to come up with the reason why SAN captured even more word order information in machine translation task. Yin et al. (2017) and Tran et al. (2018) found that the approach with a recurrence structure (e.g. RNN) has an easier time learning syntactic information than that of models with a parallel structure (e.g. CNN, SAN). Inspired by their findings, we argue that SAN tries to partially countervail its disadvantage in parallel structure by reserving more word order information, thus to help for the encoding of deeper

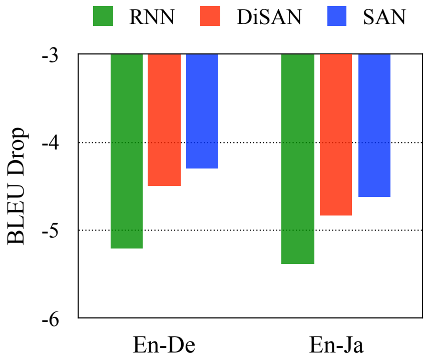


Figure 6: The differences of translation performance when the pre-trained NMT models are fed with the original ("Golden") and reordered ("Reorder") source sentences. As seen, SAN and DiSAN perform better on handling noises in terms of erroneous word order.

linguistic properties required by machine translation. Recent studies on multi-layer learning shown that different layers tend to learn distinct linguistic information (Peters et al., 2018; Raganato and Tiedemann, 2018; Li et al., 2019). The better accuracy achieved by SAN across layers indicates that SAN indeed tries to preserve more word order information during the learning of other linguistic properties for translation purpose.

Effect of Wrong Word Order Noises For humans, a small number of erroneous word orders in a sentence usually does not affect the comprehension. For example, we can understand the meaning of English sentence "Dropped the boy the ball.", despite its erroneous word order. It is intriguing whether NMT model has the ability to tackle the wrong order noises. As a results, we make erroneous word order noises on English-German development set by moving one word to another position, and evaluate the drop of the translation quality of each model. As listed in Figure 6, SAN and DiSAN yield less drops on translation quality than their RNN counterpart, demonstrating the effectiveness of self-attention on ablating wrong order noises. We attribute this to the fact that models (e.g. RNN-based models) will not learn to be robust to errors since they are never observed (Sperber et al., 2017; Cheng et al., 2018). On the contrary, since SAN-based NMT encoder is good at recognizing and reserving anomalous word order information under NMT context, it may raise the ability of decoder on handling noises occurred in the training set, thus to be more robust in translating sentences with anomalous word order.

5 Related Work

Exploring Properties of SAN SAN has yielded strong empirical performance in a variety of NLP tasks (Vaswani et al., 2017; Tan et al., 2018; Li et al., 2018; Devlin et al., 2019). In response to these impressive results, several studies have emerged with the goal of understanding SAN on many properties. For example, Tran et al. (2018) compared SAN and RNN on language inference tasks, and pointed out that SAN is weak at learning hierarchical structure than its RNN counterpart. Moreover, Tang et al. (2018) conducted experiments on subject-verb agreement and word sense disambiguation tasks. They found that SAN is good at extracting semantic properties, while underperforms RNN on capturing long-distance dependencies. This is in contrast to our intuition that SAN is good at capturing long-distance dependencies. In this work, we focus on exploring the ability of SAN on modeling word order information.

Probing Task on Word Order To open the black box of networks, probing task is used as a first step which facilitates comparing different models on a much finer-grained level. Most work has focused on probing fixed-sentence encoders, e.g. sentence embedding (Adi et al., 2017; Conneau et al., 2018). Among them, Adi et al. (2017) and Conneau et al. (2018) introduced to probe the sensitivity to legal word orders by detecting whether there exists a pair of permuted word in a sentence by giving its sentence embedding. However, analysis on sentence encodings may introduce confounds, making it difficult to infer whether the relevant information is encoded within the specific position of interest or rather inferred from diffuse information elsewhere in the sentence (Tenney et al., 2019). In this study, we directly probe the token representations for word- and phrase-level properties, which has been widely used for probing token-level representations learned in neural machine translation systems, e.g. part-of-speech, semantic tags, morphology as well as constituent structure (Shi et al., 2016; Belinkov et al., 2017; Blevins et al., 2018).

6 Conclusion

In this paper, we introduce a novel *word reordering detection* task which can probe the ability of a model to extract word order information. With the help of the proposed task, we evaluate RNN, SAN and DiSAN upon Transformer framework to empirically test the theoretical claims that SAN lacks the ability to learn word order. The results reveal that RNN and DiSAN exactly perform better than SAN on extracting word order information in the case they are trained individually for our task. However, there is no evidence that SAN learns less word order information under the machine translation context.

Our further analyses for the encoders pretrained on the NMT data suggest that 1) the learning objective sometimes plays a crucial role on learning a specific feature (e.g. word order) in a downstream NLP task; 2) modeling recurrence is universally-effective to learn word order information for SAN; and 3) RNN is more sensitive on erroneous word order noises in machine translation system. These observations facilitate the understanding of different tasks and model architectures in finer-grained level, rather than merely in overall score (e.g. BLEU). As our approach is not limited to the NMT encoders, it is also interesting to explore how do the models trained on other NLP tasks learn word order information.

Acknowledgments

The work was partly supported by the National Natural Science Foundation of China (Grant No. 61672555), the Joint Project of Macao Science and Technology Development Fund and National Natural Science Foundation of China (Grant No. 045/2017/AFJ) and the Multi-Year Research Grant from the University of Macau (Grant No. MYRG2017-00087-FST). We thank the anonymous reviewers for their insightful comments.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained Analysis of Sentence Embeddings Using Auxiliary Prediction Tasks. In *ICLR*.

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep Variational Information Bottleneck. In *ICLR*.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do Neural Machine Translation Models Learn about Morphology? In *ACL*.

Terra Blevins, Omer Levy, and Luke Zettlemoyer. 2018. Deep RNNs Encode Soft Hierarchical Syntax. In *ACL*.

Laura Bosch and Núria Sebastián-Gallés. 2001. Evidence of Early Language Discrimination Abilities in Infants from Bilingual Environments. *Infancy*, 2(1):29–49.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, Goerge Foster, Llion Jones, Parmar Niki, Mike Schuster, Zhifeng Chen, Yonghui Wu, and Macduff Hughes. 2018. The Best of Both Worlds: Combining Recent Advances in Neural Machine Translation. In *ACL*.

Yong Cheng, Zhaopeng Tu, Fandong Meng, Junjie Zhai, and Yang Liu. 2018. Towards Robust Neural Machine Translation. In *ACL*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. In *EMNLP*.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What You Can Cram into A Single $&!#∗ Vector: Probing Sentence Embeddings for Linguistic Properties. In *ACL*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL*.

Xinya Du and Claire Cardie. 2017. Identifying Where to Focus in Reading Comprehension for Neural Question Generation. In *EMNLP*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. In *ICML*.

Jie Hao, Xing Wang, Baosong Yang, Longyue Wang, Jinfeng Zhang, and Zhaopeng Tu. 2019. Modeling Recurrence for Transformer. In *NAACL*.

Hideki Isozaki, Tsutomu Hirao, Kevin Duh, Katsuhito Sudoh, and Hajime Tsukada. 2010. Automatic Evaluation of Translation Quality for Distant Language Pairs. In *EMNLP*.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR*.

Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2016. Rationalizing Neural Predictions. In *EMNLP*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-Head Attention with Disagreement Regularization. In *EMNLP*.

Jian Li, Baosong Yang, Zi-Yi Dou, Xing Wang, Michael R. Lyu, and Zhaopeng Tu. 2019. Information Aggregation for Multi-Head Attention with Routing-by-Agreement. In *NAACL*.

Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio. 2017. A Structured Self-attentive Sentence Embedding. In *ICLR*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *EMNLP*.

Makoto Morishita, Jun Suzuki, and Masaaki Nagata. 2017. NTT Neural Machine Translation Systems at WAT 2017. In *WAT*.

Graham Neubig, Yosuke Nakata, and Shinsuke Mori. 2011. Pointwise Prediction for Robust, Adaptable Japanese Morphological Analysis. In *ACL*.

Ankur Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A Decomposable Attention Model for Natural Language Inference. In *EMNLP*.

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep Contextualized Word Representations. In *NAACL*.

Alessandro Raganato and Jörg Tiedemann. 2018. An Analysis of Encoder Representations in Transformer-Based Machine Translation. In *EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*.

Benjamin Schumacher and Michael A Nielsen. 1996. Quantum Data Processing and Error Correction. *Physical Review A*, 54(4):2629.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *ACL*.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-Attention with Relative Position Representations. In *NAACL*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Shirui Pan, and Chengqi Zhang. 2018a. DiSAN: Directional Self-attention Network for RNN/CNN-free Language Understanding. In *AAAI*.

Tao Shen, Tianyi Zhou, Guodong Long, Jing Jiang, Sen Wang, and Chengqi Zhang. 2018b. Reinforced Self-attention Network: A Hybrid of Hard and Soft Attention for Sequence Modeling. In *IJCAI*.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does String-Based Neural MT Learn Source Syntax? In *EMNLP*.

Matthias Sperber, Jan Niehues, and Alex Waibel. 2017. Toward Robust Neural Machine Translation for Noisy Input Sequences. In *IWSLT*.

Emma Strubell, Patrick Verga, Daniel Andor, David Weiss, and Andrew McCallum. 2018. Linguistically-Informed Self-Attention for Semantic Role Labeling. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *ACL*.

Zhixing Tan, Mingxuan Wang, Jun Xie, Yidong Chen, and Xiaodong Shi. 2018. Deep Semantic Role Labeling with Self-attention. In *AAAI*.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why Self-Attention? A Targeted Evaluation of Neural Machine Translation Architectures. In *EMNLP*.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Sam Bowman, Dipanjan Das, et al. 2019. What Do You Learn from Context? Probing for Sentence Structure in Contextualized Word Representations. In *ICLR*.

Naftali Tishby and Noga Zaslavsky. 2015. Deep Learning and The Information Bottleneck Principle. In *ITW*.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The Importance of Being Recurrent for Modeling Hierarchical Structure. In *EMNLP*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *NIPS*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer Networks. In *NIPS*.

Shuohang Wang and Jing Jiang. 2017. Machine Comprehension Using Match-LSTM and Answer Pointer. In *ICLR*.

Zhen Xu, Bingquan Liu, Baoxun Wang, SUN Chengjie, Xiaolong Wang, Zhuoran Wang, and Chao Qi. 2017. Neural Response Generation via GAN with An Approximate Embedding Layer. In *EMNLP*.

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling Localness for Self-Attention Networks. In *EMNLP*.

Baosong Yang, Longyue Wang, Derek F. Wong, Lidia S. Chao, and Zhaopeng Tu. 2019. Convolutional Self-Attention Networks. In *NAACL*.

Baosong Yang, Derek F Wong, Tong Xiao, Lidia S Chao, and Jingbo Zhu. 2017. Towards Bidirectional Hierarchical Representations for Attention-based Neural Machine Translation. In *EMNLP*.

Wenpeng Yin, Katharina Kann, Mo Yu, and Hinrich Schütze. 2017. Comparative Study of CNN and RNN for Natural Language Processing. *arXiv preprint:1702.01923*.

Adams Wei Yu, Hongrae Lee, and Quoc Le. 2017. Learning to Skim Text. In *ACL*.